COSC603 Project 1

Name: Tao Qiu

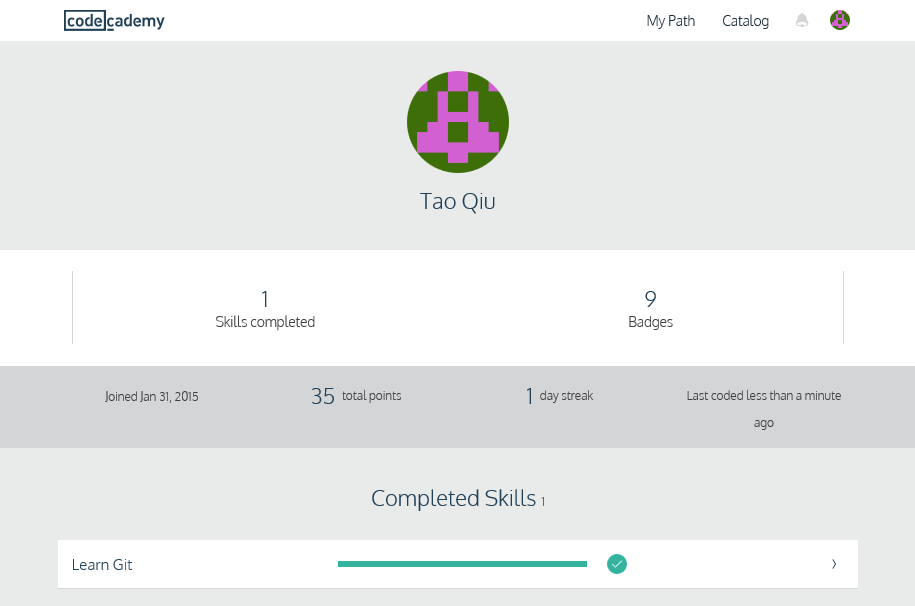
Task1

Since I already have Eclipse installed in my computer, I just went ahead and installed the Javadoc and JAutodoc according to the instructions.

I wrote a small HelloWorld program and some words as comments and generate a Javadoc. It worked fine. A fold with all the documents was generated. Actually, in the past I have read lots of Javadoc, and thought people purposely wrote them. Now I know it can be automatically generated. Great! The new thing I figure out is that I should use a correct and standardized naming styles for classes and methods in the future. In that case I could get better documents.

I also have setup a GitHub account. It works well.

Task 3:



I have completed the training in Codecademy. The achievement is shown above. Basically, I feel this is a very good training course for me because I just have little knowledge about the Git. I have learn some basic operations and commands in Git such as creating new branch, deleting the branch, adding to new stage, unstaging, committing the branch, merging the branch with master, setting up the origin branch and merging the origin/master, pushing to and fetching from remote. In general, what I learn are all the Git commands. They are fast, easy to use, but I have to memorize them. In some complex situation, it is hard to find a good command to use, therefore I would rather to use a Git GUI to work on Github. I have downloaded and installed the GitHub Desktop on my computer.

Eventually, I figured out the EGit, rather than GitHub Desktop, is the best choice for me to connect Github, because the EGit directly work with Eclipse and integrated into the Eclipse. Why not just use EGit, so I have added the EGit plugin into Eclipse from the Eclipse marketplace. Even though there some issues after installation, I have solved them and was able to push my project “FileDangerCalculation” and a test program “HelloWorld” to the Github. I also did several updates and committed and pushed again. The Github worked well.

Task 5:

By analyzing the Fire Danger program, I realized it is in a format of very old fashion. The program was run in only one thread up down, using “goto” statements to dump here and there. There is no separated functions or methods. The program routes cannot be identified easily, so it’s hard to understand their logic. However, the program provides lots of good comments and explains, with diagram showing their working flow. Unfortunately, the working flow does not match their codes very well, just around 85% of working flow matched the codes.

To reformat it to a Java format, and I have established a project called “FileDangerCalculation” , in which there is a class called “Test”. “Test” contains a main method to run the program. In the “test” class I placed 4 steps to run the test. 1) getting the inputted values. 2) getting the table values. 3) calculating updated parameters(getDanger) . 4) output the result. The program works.

The experience I learn from this excise is that: 1) We should be careful with program structure. It should contains the modularity with levels of nesting; 2) We should have a good naming style. Just in this practice I started to know the difference between “/\*” and “//”. They are actually different. The text following “/\*” will be extracted into the Javadoc for the information for the class; 3) We should use a standardized way to present the code including the good use of indentation and space. 4) Specifically we need to change the folder for Javadoc to local master folder of Git after installation of EGit, otherwise the doc cannot be updated.

The document generated by Javadoc is placed at Github: [COSC603-qiu-project1](https://github.com/tqiu1/COSC603-tqiu-project1)/[FireDangerCalculation](https://github.com/tqiu1/COSC603-tqiu-project1/tree/master/FireDangerCalculation)/**experiences**/

The result of program output with random inputs is shown as below:

